PRACTICAL NO. 2: Remote Method Invocation

1. To implement arithmetic calculator using RMI concept. Server will provide calculator functions and client send requests to server for specific functions of calculator.

Code:

Calculator.java

```
package calc;
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface Calculator extends Remote
{
double add(double a, double b) throws RemoteException;
double sub(double a, double b) throws RemoteException;
double mul(double a, double b) throws RemoteException;
double div(double a, double b) throws RemoteException;
}
```

CalculatorImpl.java

```
package calc;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class CalculatorImpl extends UnicastRemoteObject implements Calculator
{
	protected CalculatorImpl() throws RemoteException
	{
		super();
	}
	@Override
	public double add(double a, double b) throws RemoteException {
		return a+b;
	}
	@Override
	public double sub(double a, double b) throws RemoteException {
		return a-b;
	}
	@Override
	public double mul(double a, double b) throws RemoteException {

		return a*b;
	}
	@Override
	public double div(double a, double b) throws RemoteException {
		return a/b;
	}
}
```

CalculatorSever.java

```
package calc;
```

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class CalculatorServer {
	public static void main(String[] args)
	{
		try {
			Calculator calculator=new CalculatorImpl();
			Registry registry = LocateRegistry.createRegistry(1099);
			registry.rebind("CalculatorService", calculator);
			System.out.println("Calculator RMI Server is running....");

		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

CalculatorClient.java

```
package calc;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Scanner;
public class CalculatorClient {
	public static void main(String[] args) {
		try {
			Registry registry =LocateRegistry.getRegistry("localhost",1099);
			Calculator calculator =(Calculator) registry.lookup("CalculatorService");

			Scanner sc = new Scanner(System.in);
			System.out.println("Enter a First number: ");
			double a = sc.nextDouble();
			System.out.println("Enter a  Second Number");
			double b = sc.nextDouble();

			System.out.println("Choose operation (+,-,*,/): ");
			char op =sc.next().charAt(0);

			double result =0;
		switch (op) {
		    case '+':
		       result = calculator.add(a, b);
		       System.out.println("Result: " + result);
		       break;
		    case '-':
		       result = calculator.sub(a, b);
		       System.out.println("Result: " + result);
		       break;
		    case '*':
```

```
                    result = calculator.mul(a, b);
                    System.out.println("Result: " + result);
                    break;
                case '/':
                    if (b == 0) {
                        System.out.println("Cannot divide by zero!");
                    } else {
                        result = calculator.div(a, b);
                        System.out.println("Result: " + result);
                    }
                    break;
                default:
                    System.out.println("Invalid operation.");
            }
            } catch (Exception e) {
                    e.printStackTrace();
            }
        }
}
```

Output:

```
Calculator RMI Server is running....
```

```
Enter a First number:
10
Enter a  Second Number
5
Choose operation (+,-,*,/):
+
Result: 15.0
```

2. Retrieve day, time and date function from server to client. This program should display server day, date and time. (Use Concept of RMI)

Code:
DateTimeService.java

```
package DateTime;
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface DateTimeService extends Remote {
        String getDate() throws RemoteException;
  String getTime() throws RemoteException;
```

```
}
```

DateTimeServiceImpl.java

```
package DateTime;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
public class DateTimeServiceImpl extends UnicastRemoteObject implements DateTimeService  {
            protected DateTimeServiceImpl() throws RemoteException {
      super();
   }
   @Override
   public String getDate() throws RemoteException {
      LocalDate date = LocalDate.now();
      return date.format(DateTimeFormatter.ofPattern("yyyy-MM-dd"));
   }
   @Override
   public String getTime() throws RemoteException {
      LocalTime time = LocalTime.now();
      return time.format(DateTimeFormatter.ofPattern("HH:mm:ss"));
   }
}
```

DateTimeServer.java

```
package DateTime;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class DateTimeServer {
   public static void main(String[] args) {
      try {
         DateTimeService dateTimeService = new DateTimeServiceImpl();
         Registry registry = LocateRegistry.createRegistry(2000);
         registry.rebind("DateTimeService", dateTimeService);
         System.out.println("DateTime RMI Server is running...");
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

DateTimeClient.java

```
package DateTime;
```

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class DateTimeClient {
  public static void main(String[] args) {
     try {
        Registry registry = LocateRegistry.getRegistry("localhost", 2000);
        DateTimeService dateTimeService = (DateTimeService) registry.lookup("DateTimeService");
        String date = dateTimeService.getDate();
        String time = dateTimeService.getTime();
        System.out.println("Server Date: " + date);
        System.out.println("Server Time: " + time);
     } catch (Exception e) {
        e.printStackTrace();
     }
  }
}
```

output:

```
DateTime RMI Server is running...
```

```
Server Date: 2025-09-03
Server Time: 13:23:00
```

3. Equation solver. The client should provide an equation to the server through an interface. The server will solve the expression given by the client. $(a-b)^2 = a^2 - 2ab + b^2$; If a = 5 and b = 2 then return value = $5^2 - 2*5*2 + 2^2 = 9$.

Code:

EquationSolver.java

```
package rmiEquation;
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface EquationSolver extends Remote {
        double solveEquation(double a, double b) throws RemoteException;
}
```

EquationSolverImpl.java

```
package rmiEquation;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
```

```
public class EquationSolverImpl extends UnicastRemoteObject implements EquationSolver {
        protected EquationSolverImpl() throws RemoteException {
    super();
  }
        @Override
        public double solveEquation(double a, double b) throws RemoteException {
                // TODO Auto-generated method stub
                return (a*a) - (2*a*b) + (b*b);
        }
}
```

EquationServer.java

```
package rmiEquation;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class EquationServer {
  public static void main(String[] args) {
    try {
      EquationSolverImpl solver = new EquationSolverImpl();
      Registry registry = LocateRegistry.createRegistry(1098); // default RMI port
      registry.rebind("EquationService", solver);
      System.out.println("Equation Solver RMI Server is running...");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

EquationClient.java

```
package rmiEquation;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Scanner;
public class EquationClient {
  public static void main(String[] args) {
    try {
      Registry registry = LocateRegistry.getRegistry("localhost", 1098);
      EquationSolver solver = (EquationSolver) registry.lookup("EquationService");
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter value of a: ");
      double a = sc.nextDouble();
      System.out.print("Enter value of b: ");
      double b = sc.nextDouble();
      double result = solver.solveEquation(a, b);
```

```
            System.out.println("Result of (a-b)^2 = " + result);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Equation Solver RMI Server is running...
```

```
Enter value of a: 5
Enter value of b: 2
Result of (a-b)^2 = 9.0
```

Conclusion:

In this practical, we successfully implemented Java RMI programs for calculator, date-time, and equation solver applications.RMI allowed remote objects to be accessed as if they were local, hiding the complexity of network communication.This enhanced our understanding of distributed programming, remote method invocation, and client-server interaction in Java.